Motivated Reinforcement Learning for Adaptive Characters in Open-Ended Simulation Games

Kathryn Merrick
School of Information Technologies,
University of Sydney and
National ICT Australia
+61 2 8374 5590
kkas0686@it.usyd.edu.au

Mary Lou Maher
Key Centre for Design Computing
and Cognition,
University of Sydney
mary@arch.usyd.edu.au

ABSTRACT
Recently a new generation of virtual worlds has emerged in which users are provided with open-ended modelling tools with which they can create and modify world content. The result is evolving virtual spaces for commerce, education and social interaction. In general, these virtual worlds are not games and have no concept of winning, however the open-ended modelling capacity is nonetheless compelling. The rising popularity of open-ended virtual worlds suggests that there may also be potential for a new generation of computer games situated in open-ended environments. A key issue with the development of such games, however, is the design of non-player characters which can respond autonomously to unpredictable, open-ended changes to their environment. This paper considers the impact of open-ended modelling on character development in simulation games. Motivated reinforcement learning using context-free grammars is proposed as a means of representing unpredictable, evolving worlds for character reasoning. This technique is used to design adaptive characters for the Second Life virtual world to create a new kind of open-ended simulation game.

Categories and Subject Descriptors
I.2.8 [**Artificial Intelligence**]: Learning – *neural nets,* I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search – *dynamic programming, heuristic methods.*

General Terms
Algorithms.

Keywords
Motivated reinforcement learning, context-free grammar, adaptive characters, computer games.

ACE'07, June 13–15, 2007, Salzburg, Austria.

1. INTRODUCTION
Simulation games such as The Sims and organic simulation games such as Creatures are distinguished by characters that can respond to certain changes in their environment with new behaviours. Human players modify the circumstances that surround each non-player character (NPC) in order to influence the emergence of certain types of character behaviour for which points may be awarded or which trigger a new phase of the game. Existing simulation and organic simulation games are, however, limited by the set of changes which players may make to the game environment while the game is in progress.

In contrast, human users in current virtual worlds such as Second Life can use open-ended modelling tools to create and modify world content in a manner that is relatively unrestricted when compared to existing game worlds. The popularity and rapid growth in the user base of open-ended virtual worlds suggests that there is potential for a new generation of computer game situated in open-ended environments. A key challenge to be overcome in the development of such games, however, is the development of characters which can respond autonomously to the unpredictable, open-ended changes in their environment. This paper considers the impact of open-ended modelling on character development in simulation games. The need is identified for new kinds of character control algorithms which can adapt and respond to unpredictable changes in the game environment without advance knowledge of what these changes may be or when they may occur. Motivated reinforcement learning (MRL) using context-free grammars (CFGs) is proposed as a means of representing unpredictable, evolving worlds for character reasoning.

Sections 2 and 3 of this paper review existing open-ended virtual worlds and simulation games. Existing character algorithms for simulation games are discussed and the need for new, adaptive control algorithms is identified for open-ended simulation games. Section 4 proposes MRL using CFGs as an approach to the design of adaptive characters. CFGs offer a means of representing unpredictable, evolving worlds for character reasoning while the combination of motivation and reinforcement learning enable characters to identify and learn about changes in their environment. Section 5 demonstrates the design of adaptive characters to create an open-ended simulation game in the Second Life virtual world.

2. OPEN-ENDED VIRTUAL WORLDS
The earliest open-ended virtual worlds were text-based, object oriented, multi-user dungeons (MOOs = MUDs, object oriented). MOOs are persistent, multi-user, interactive systems which can be thought of as low-bandwidth virtual worlds. MOOs such as

LambdaMOO [10] are distinguished from MUDs by the ability for users to perform object oriented programming within the MOO server, expanding and changing how the MOO server behaves to all users. Examples of changes include authoring new rooms and objects and changing the way the MOO interface operates. These changes are made using a MOO programming language which often features libraries of verbs that can be used by programmers in their coding.

More recently, the improvement in computer graphics technologies has made large scale, 3D virtual worlds possible. Two current examples of open-ended virtual worlds are Second Life [4] and Active Worlds [1]. Screen shots from these worlds, including their 3D modelling tools, are shown in Figure 1 and Figure 2. In Second Life and Active Worlds, the virtual landscape can be extended using a combination of primitive shapes and textures to create new buildings, plants, animals and other artefacts. The ability to define object geometry and to assign natural language labels and descriptions to designed objects are key aspects which distinguish these virtual worlds as open-ended environments.

While the users of MOOs and 3D virtual worlds have used their open-ended expansion capacities to develop games within these environments, these games have tended to adhere to existing genres such as adventure and role-playing games. The development of games which incorporate and respond to the open-ended building capabilities of virtual worlds has been difficult due to the lack of character control techniques which can operate in such environments. This paper considers how MRL agents can be used to control characters in open-ended games using CFGs to represent the changing environment. In particular, we focus on simulation games, reviewed in the next section.

Figure 1. In the Second Life virtual world, complex environments can be created using a combination of primitives and uploaded textures.

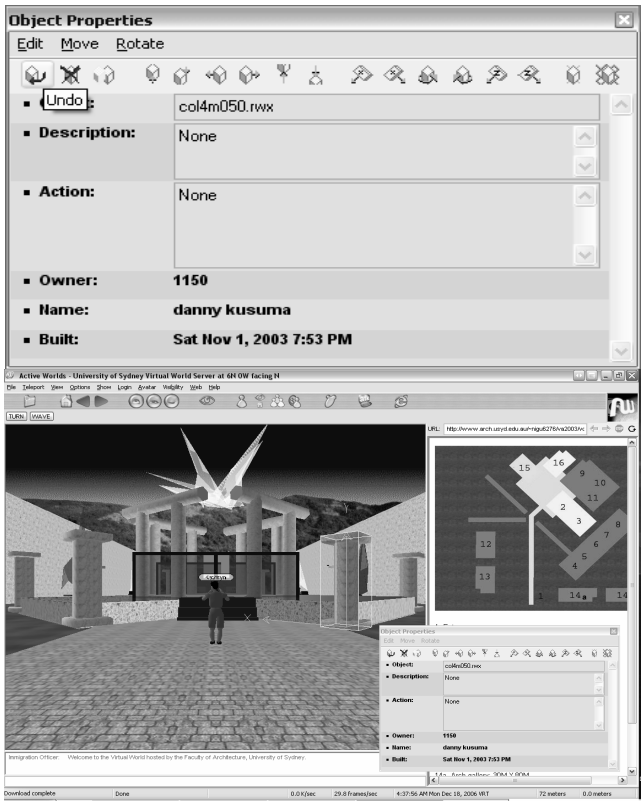

Figure 2. In Active Worlds, complex scenes can be created using basic shapes and textures.

3. SIMULATION GAMES

Simulation games such as The Sims and organic simulations games such as Black and White and Creatures give a human player control over a simulated world. Players can modify the game environment and interact with its NPC inhabitants in certain ways and then observe the effects of actions on individual characters and the virtual society [3]. We identify these games as a starting point for making new games with more open ended, in-game modification capabilities as limited in-game world modification is already a key aspect of this game genre. The key question addressed in this paper is how to model characters which can learn to respond to open-ended modifications through interaction with the objects in the world.

Existing artificial intelligence technologies for controlling NPCs in simulation games range from reflex agents to learning and evolutionary algorithms. A key paradigm to arise from simulation games is the smart terrain concept developed by Will Wright for The Sims. While these techniques create NPCs which can adapt to certain types of changes in their environment, we argue they do not extend to autonomous, adaptive behaviour in open-ended environments. In the following sections, we discuss the issues with these techniques with respect to open-ended environments and consider MRL as an alternative.

3.1 Reflex Agents

Reflexive behaviour is a pre-programmed response to the state of the environment – a reflex without reasoning [5]. Only recognised

states will produce a response. One example of a reflex agent technology used in simulation games is the fuzzy state machine paradigm used in the Sims. Fuzzy state machines combine state machine and fuzzy logic technologies to create agents which can identify and respond to states which are within some decision threshold of a predefined condition [2]. For example, where a NPCs controlled by a state machine may consider balls as a target for kicking, NPCs controlled by fuzzy state machines may consider any object which, within some threshold, fits the description of 'being round' as a target for kicking. Characters with different personalities can be defined by building fuzzy state machines with different decision thresholds.

While fuzzy state machines have been used with success in existing simulation games, the need to define states and thresholds before the character is introduced to its environment limits the character to action within the predefined boundaries of states and decision thresholds. This technology thus becomes problematic in environments which can be modified in an open-ended manner.

3.2 Smart Terrain

A key paradigm to arise from simulation games is the smart terrain concept developed for The Sims [12]. Smart terrain discards the character-oriented approach to reasoning using artificial intelligence and embeds the behaviours and possible actions associated with a virtual object within the object itself. For example, the file for the model of a television in The Sims might contain the instructions for watching it, turning it on and off, the conditions under which a Sim might want to watch it and how a Sim should be animated while watching it. This approach allows new objects which an be inserted into the game at any point, either as an expansion pack by game designers or using content creation tools by players. Achieving adaptability using this approach, however, requires character behaviours to be explicitly programmed in each new object. This requires development effort from game designers and, while compelling for some gamers, is too difficult or simply uninteresting for others.

Expansion packs and content creation tools in general are approaches by which game designers have attempted to extend the lifetime of games by extending or allowing players to extend the original game through the addition of new content. Games in which open-ended modification of the game world is allowed while the game is in progress has the potential for a longer lifetime though the provision of more open-ended game play.

3.3 Learning Agents

Learning agents are able to modify their internal structure in order to improve their performance with respect to some task [9]. In some games such as Black and White, NPCs can be trained to learn behaviours specified by their human master. The human provides the NPC with feedback such as food or patting to encourage desirable behaviour and punishment to discourage unwanted actions. While the behaviour of these characters may potentially evolve in any direction desired by the human, behaviour development relies on feedback from human players. For autonomous learning in simulation games in open-ended environments, learning techniques are also required which allow characters to learn without direct supervision from players.

4. MOTIVATED REINFORCEMENT LEARNING USING CONTEXT-FREE GRAMMARS

In MRL, computational models of motivation focus reinforcement learning (RL) [11] to support the emergence of rhythmic behavioural cycles, adaptive behaviour and multi-task learning. RL agents learn by trial and error through direct interaction with their environment. Because MRL is based on RL, it offers a means of autonomous learning for NPCs which does not require direct supervision by human player characters. The motivation component which is based on task independent concepts such as novelty or interest, is general enough to direct learning towards different tasks which might arise without requiring prior knowledge of what those tasks may be or when they should be performed. The problem remains of how to represent environments which may change over time in order to apply MRL. In this section we propose the use of CFGs as an approach to this representation.

In the RL formalism, the environment in which learning takes place is defined as a Markov decision process P in which a learning agent senses some subset **S** of the environment, reasons about the sensed world and uses a set **A** of actions to trigger effectors which perform some task. Game environments, however, are made complex by the presence of multiple tasks which may change over time. In these environments it is frequently the case that only subsets of the state and action spaces are relevant to a given task at a particular time. In this paper, we model such complex, dynamic environments as a set of MDPs, $\mathbf{P} = \{P_1, P_2, P_3 \ldots\}$, in which each P_i represents a region of the environment in which a task is situated. P_i comprises a set $\mathbf{S}_i$ of states, a set $\mathbf{A}_i$ of actions and a task defined by a reward function $\mathcal{R}_i$. We define the complexity of the environment in terms of the number of elements in the set **P**. An environment is more complex if the set of MDPs representing it contains more elements. We define an environment to be dynamic if the elements in **P** change over time, either through the addition of new elements or the removal of existing elements.

In order for a RL agent to learn to perform a task in an environment modelled by a set **P** of MDPs, a reward signal $\mathcal{R}_i$ is required which represents that task. Traditionally, reward signals have been defined prior to a learning agent interacting with its environment, using a rule based representation which comprises a set of conditions about subsets of its state space or the transitions between states. However, life-long learning required by agents in complex, dynamic environments such as game worlds is characterised not by the learning of a single task but by adaptive learning which focuses on different tasks at different times. Learning which focuses attention on more than one task, requires a reward signal which represents the union of several reward functions $\mathcal{R}_U = \mathcal{R}_a \cup \mathcal{R}_b \cup \mathcal{R}_c \ldots$. Likewise learning adaptive behaviours by focusing attention on different tasks at different times, requires a reward signal that represents a progression between different reward signals $\mathcal{R}_P = \mathcal{R}_a \rightarrow \mathcal{R}_b \rightarrow \mathcal{R}_c \ldots$. The design of reward signals which represent the union of or progression between reward signals for different tasks using a rule based approach becomes problematic in complex or dynamic environments. In complex environments designing reward functions for each element of **P** is time consuming and requires prior knowledge of large numbers of subsets of the state or

transition space to develop the rules which will define each task. In dynamic environments the reward functions designed may become obsolete if the tasks they define are no longer possible, while focus may never be given to new tasks which arise.

The role of motivation in MRL is to provide an approach to the design of reward signals which approximates the union of or progression between reward functions for different tasks without the need to identify subsets of the state or transition space prior to learning to compose rules which will define each task. This extends RL beyond single task learning to life-long, adaptive, multi-task learning in complex, dynamic environments. The environment model for MRL in complex, dynamic environments becomes:

- a set $\mathbf{S} = \mathbf{S}_1 \cup \mathbf{S}_2 \cup \mathbf{S}_3 \cup \ldots$ of sensed states,
- a set $\mathbf{A} = \mathbf{A}_1 \cup \mathbf{A}_2 \cup \mathbf{A}_3 \cup \ldots$ of actions,
- a motivation function $\mathcal{M}: \mathbf{S} \times \mathbf{A} \rightarrow \{\text{Reals}\}$
- a transition function $\mathcal{P}: \mathbf{S} \times \mathbf{A} \rightarrow \pi(\mathbf{S})$

In this model, the motivation function $\mathcal{M}(S_{(t)}, A_{(t)}, S_{(t+1)})$ is the expected motivation for a transition $T_{(t+1)}$ from $S_{(t)}$ to $S_{(t+1)}$ when $A_{(t)}$ is executed. $\mathcal{M}(S_{(t)}, A_{(t)}, S_{(t+1)})$ is an approximation of the progression between reward signals for one or more tasks. The aim of the motivation function is to produce motivation values which will motivate the RL process to focus on executing actions which support the emergence of focused behavioural cycles in the character actions.

In dynamic environments, the traditional, fixed length vector representation for S becomes inadequate as a fixed length representation cannot represent changes in the state space when the environment changes through the addition or removal of elements P_i. As an alternative to fixed length vectors, we propose that sensed states $S_{(t)}$ may be represented as a string from a CFG [6] ($\mathbf{V}_S$, $\mathbf{\Gamma}_S$, $\mathbf{\Psi}_S$, S) where:

- $\mathbf{V}_S$ is a set of variables or syntactic categories,
- $\mathbf{\Gamma}_S$ is a finite set of terminals such that $\mathbf{V}_S \cap \mathbf{\Gamma}_S = \{\}$,
- $\mathbf{\Psi}_S$ is a set of productions of the form V → v where V is a variable and v is a string of terminals and variables,
- S is the start symbol.

Thus, the general form of a sensed state is:

```
S              → <sensations>
<sensations>   → <PiSensations><sensations> | ε
<PiSensations> → <sL><PiSensations> | ε
<sL>           → <number> | <string>
<number>       → 1 | 2 | 3 | ...
<string>       → ...
```

This representation is flexible enough to represent environments containing different numbers of elements $\mathbf{P}_i$. This is important in dynamic game worlds where the elements comprising **P** may change over time as players modify the game environment by building or crafting new objects. This representation is also flexible enough to represent objects with different properties. For example, one object may have a shape and colour, while another may also have an age or usage limit and so on.

In fixed length vector representations two states can be compared by comparing the values of vector elements with the same index in each state. In variable length sensed states, a label L is assigned to each sensation by the sensor which produced it, such that two states can be compared by comparing the values of sensations that have the same label where such sensations exist.

A flexible representation is also required for the action space in dynamic environments. While characters may maintain a fixed set of effectors for the duration of their life, the actions they can perform with those effectors may change with the addition or removal of elements P_i from the environment. We also represent the action space **A** using a CFG ($\mathbf{V}_A$, $\mathbf{\Gamma}_A$, $\mathbf{\Psi}_A$, **A**) where:

- $\mathbf{V}_A$ is a set of variables or syntactic categories,
- $\mathbf{\Gamma}_A$ is a finite set of terminals such that $\mathbf{V}_A \cap \mathbf{\Gamma}_A = \{\}$,
- $\mathbf{\Psi}_A$ is a set of productions of the form V → v where V is a variable and v is a string of terminals and variables,
- **A** is the start symbol.

Thus, the general form of the action space is:

```
A              → <actions>
<actions>      → <PiActions><actions>  | ε
<PiActions>    → <Aj><PiActions> | ε
<Aj>           → ...
```

The algorithm in Figure 3 represents our basic MRL control strategy for agent reasoning which achieves attention focus that supports life-long, adaptive, multi-task learning in complex, dynamic environments. We model algorithms for MRL as continuing task, temporal difference (TD) learning algorithms using generalised policy iteration.

```
1.  Repeat (forever):
2.      Sense S(t)
3.      Compute motivation Rm(t)
4.      if (Q(S(t), A) not initialised):
5.         init Q(S(t),A) arbitrarily ∀A
6.      Choose A(t) from S(t) using a policy
        improvement fn
7.      if (S(t-1) is initialised):
8.         Update Q for S(t-1) and A(t-1) using a
           policy evaluation fn
9.      S(t-1) ← S(t); A(t-1) ← A(t)
10.     Execute A(t)
```

Figure 3. The motivated reinforcement learning algorithm.

Our MRL algorithm is arranged in four phases concerned with sensing the environment (line 2), computing motivation (line 3), policy improvement and evaluation (lines 4-8) and activation (line 10). The key differences between this approach and standard RL is the introduction in line 3 of a step for computing a motivation value and the incremental initialisation of the state-action table in lines 7-8. In MRL the reward signal, traditionally defined by rules about the states space in RL, is replaced by a motivation signal computed based on experiences. Because motivation is computed based on agent experiences rather than rules defined prior to learning, the motivation signal can adapt to changes in the environment experienced by the agent and adapt the focus of learning accordingly. The state-action table or equivalent is also initialised with sensed states and actions incrementally as attention is focused on new states, rather than prior to learning.

This means that knowledge of the state and action spaces is not required prior to learning.

In this paper we compute motivation by identifying interesting events using the algorithm described by Merrick and Maher [7] and summarised in Figure 4. Events E are computed as the difference between or change in two successive sensed states. The key component of this model of motivation is an habituated self-organising map (HSOM) which clusters similar events and uses the clustering error to trigger the computation of novelty. The HSOM can be modified to accept variable length strings from a CFG as input by initialising each SOM neuron as a zero length vector. Each time a stimulus event is presented to the HSOM, each neuron is lengthened by adding randomly initialised variables with any labels L that occur in the event but not in the neuron. String valued sensations are enumerated and events normalised before input to the HSOM. The novelty value output by the HSOM is modified using the Wundt curve to produce an interest value which is used as the motivation signal $R_{m(t)}$.

1. Subtract $S_{(t)} - S_{(t')}$ using

$$\Delta(s_{L(t)}, s_{L(t-1)}) = \begin{cases} s_{L(t)} \text{ if } \neg\exists\, s_{L(t')} \\ -s_{L(t')} \text{ if } \neg\exists\, s_{L(t)} \\ s_{L(t)} - s_{L(t')} \text{ if } s_{L(t)} - s_{L(t')} \neq 0 \\ \text{null otherwise} \end{cases}$$

2. Compose $E_{(t)} = (\Delta(s_{1(t)}, s_{1(t-1)}), \Delta(s_{2(t)}, s_{2(t-1)}), \ldots \Delta(s_{L(t)}, s_{L(t-1)}),\ldots)$
3. Compute $N_{(t)} = \mathcal{N}(E_{(t)})$ using an HSOM.
4. Compute $I(N_{(t)})$ using

$$I(N_{(t)}) = \frac{F^{+}_{max}}{1 + e^{-\rho^{+}(2N_{(t)} - F^{+}_{min})}} - \frac{F^{-}_{max}}{1 + e^{-\rho^{-}(2N_{(t)} - F^{-}_{min})}}$$

5. Output $R_{m(t)} = I(N_{(t)})$

Figure 4. Algorithm for motivation using interesting events.

The Wundt curve peaks for moderate novelty values so the most interesting and thus most highly motivating events are those that are similar-yet-different to previously encountered experiences.

5. DEMONSTRATION: AN OPEN-ENDED SIMULATION GAME IN SECOND LIFE

In this section we apply the MRL model described above to NPCs in an organic simulation game implemented in Second Life, an open-ended virtual world. The demonstration shows how MRL agents using CFGs can respond to new objects built in their environment without requiring information about how they should respond to be programmed into the object or provided as feedback from humans.

5.1 The Game

The game we developed, shown in Figure 5, is a multi-player sheep herding game in which the aim is to modify the environment in order to attract and keep the attention of as many sheep as possible.

5.2 System Architecture

Characters can be created in Second Life by connecting an agent program written in a programming language such as Java to a 3D model of the character, in this case a sheep, using the framework shown in Figure 6. The Java agent program consists of sensor stubs, agent reasoning processes and effector stubs. The Java sensor and effector stubs act as clients which connect via XML-RPC to corresponding sensor and effector stubs written in the Linden Scripting Langauge (LSL) and residing on a Second Life server. LSL sensor and effector stubs are associated with each sheep allowing them to sense, reason about and affect their environment. This cycle repeats approximately every seven seconds.

Figure 5. A sheep herding game in Second Life. Sheep can initially sense only a tuft of eelgrass.

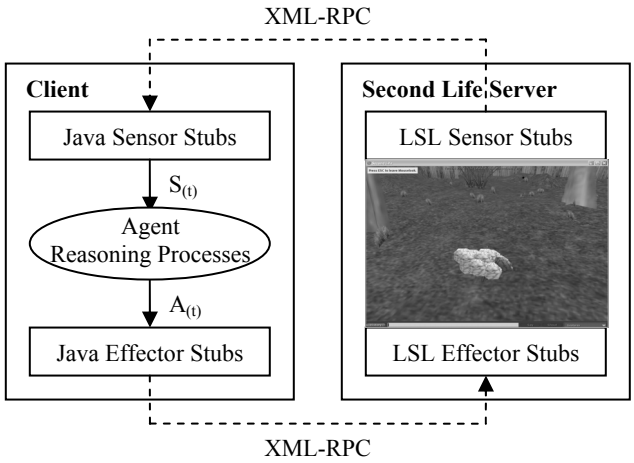

Figure 6. System architecture for designing characters for Second Life.

In our game, the sheep are MRL agents with sensors enabling them to monitor objects, avatars and their location in the world and effectors allowing them to move to or use objects and avatars:

- Object sensor: monitors the environment and returns data describing objects within a six metre radius of the sheep. Data includes the name and quantity of the object. Sheep cannot sense other sheep[1].
- Avatar sensor: monitors the environment and returns data describing avatars within a six metre radius of the sheep. Data includes the name of the avatar.

[1] When sheep can sense each other, the emergent learned behaviour resembles flocking. While an interesting phenomenon, this flocking makes it more difficult for players to attract the attention of sheep. It could perhaps be used as an "Advanced" game setting however.

- Location sensor: monitors the environment and returns a sheep's location as the id of closest object within a three metre radius or NOWHERE otherwise.
- Move to object effector: moves this sheep to the given object within a six metre radius by applying a constant force over one second.
- Jump to object effector: moves this sheep to the given object with a four to six metre radius by applying an upwards impulse and a directional force over one second.
- Use object effector: triggers a behaviour to be executed by the given object. Object behaviours may include any action scripted by a player, such as a change in the object or the creation of new objects and so on.

While the set of sensors and effectors is fixed, the sensations they may return and the effects they may have on the environment are not. There is, for example, no predefined set of objects or avatars which may be sensed. Any player avatar may build and uniquely name any object to attempt to attract the attention of a sheep. Likewise, there is no predefined set of behavioural responses a 'usable' object may have to a sheep's attempt to use it. The 'use object effector' in particular adds complexity an otherwise simple game by allowing players to express their creativity by designing objects which respond in novel, and hopefully interesting, ways to the sheep's attempts to use them. This can be seen in the game play sequences described in the next section.

5.3 Game Play Sequences

Initially, the game begins with five sheep in an empty paddock, with only a single tuft of grass. Players have available all the Second Life building and scripting tools to build new objects with which the sheep can interact, either by moving amongst or using. The following figures display some of the items built for the sheep. We describe environment representation constructed in response to these objects and the behaviours the sheep developed.

Prior to the appearance of the player avatars, sheep can sense only a tuft of grass:

$S_{(1)}$ = (<Eelgrass:3><location:Eelgrass>)

and have an action set:

$\mathbf{A}_{(1)}$ = {move to(Eelgrass), use(Eelgrass)}

Because the available sensations and the action set are limited to sensations and actions concerning the eelgrass, the sheep can only be motivated to learn about the eelgrass. They experiment with actions such as `move to(Eelgrass)` and `use(Eelgrass)` however these actions produce no events as the sheep are already at the eelgrass and the eelgrass does not trigger a scripted response when used. The appearance of new objects or avatars is thus likely to trigger the emergence of new behaviours by the sheep.

In the first game play sequence a player, represented by an avatar called Sahi Kipling, moves the avatar close to the sheep. The state sensed by one sheep changes to:

$S_{(2)}$ = ((<Eelgrass:3><location:Eelgrass><Sahi Kipling:1>)

and the action set to:

$\mathbf{A}_{(2)}$ = {move to(Eelgrass), use(Eelgrass), move to(Sahi Kipling), jump to(Sahi Kipling)}

The sensed state and action sets change in response to the changing environment, to represent the objects and avatars currently sensed by a sheep. The sheep are now able to experiment with new actions. A motivation signal is computed based on the interest of each action attempted. Some sheep continue to use the `move to(Eelgrass)` and `use(Eelgrass)` actions while another experiments with the `move to(Sahi Kipling)` and `jump to(Sahi Kipling)` actions. This produces an event indicating the sheep has changed its location:

$E_{(2)}$ = (location(1))

The player progressively moves their avatar away from the sheep so the sheep continues to have a 'move to Sahi Kipling' action in its action set which triggers a location change event when performed. Interest in this event increases according to the Wundt curve, eventually reinforcing the 'move to Sahi Kipling' action so that an emergent 'following' behaviour is learned. This allows the player to draw the sheep away from the grass towards another part of the world as show in Figure 7(a). Figure 7(b) shows the behavioural cycle learned by the sheep in response to the player.

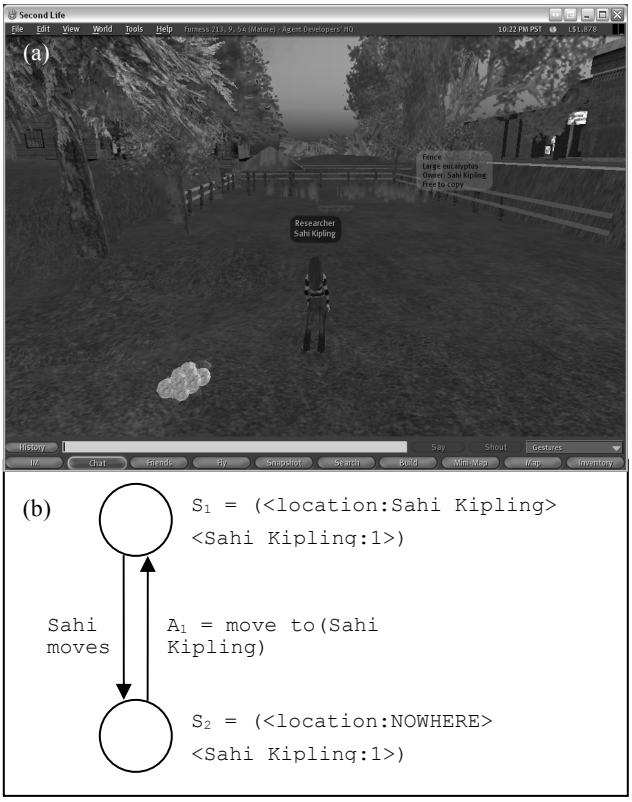

Figure 7. (a) A sheep is motivated to learn to follow an avatar away from the eelgrass. (b) The learned behavioural cycle of the sheep.

In the second game play sequence, shown in Figure 8(a), the player has drawn a sheep away from the eelgrass, built a 'food machine' in its immediate environment then run out of the sensory radius of the sheep. The food machine is scripted so that, should the sheep choose to use it, food will appear and slide down the shoot. The food, when used (eaten) will disappear. The state sensed by the sheep near the food machine in Figure 8 is:

$S_{(1)}$ = (<Food machine:1><Food:2><location:Food machine>)

and its action set is:

$\mathbf{A}_{(1)}$ = {move to(Food), jump to(Food), use(Food), move to (Food Machine), use(Food)}

This example shows how the CFG representation allows the sheep to adapt to its new setting. Only the sensations and actions relevant to its current situation are included in the sensed state and action set. Unlike fixed length representations which would be required to maintain a representation of Sahi Kipling and the eelgrass even after they are no long relevant, the CFG represents only the data currently sensed by the learning agent. In our game scenario, this allows a sheep to experiment with just the actions relevant to its current situation, and to adapt its representation when its current situation changes. As the sheep experiments with each available action, a motivation value is computed indicating the interest of the event caused by each action. These motivation values again reinforce certain actions, focus learning and stimulating the emergence of new behaviours.

Because motivation is based on experience and the experience of each sheep is different, different behavioural responses to the same situation are possible. In the case of the food machine, for example, we observed that different sheep developed different behavioural responses to the machine over time. Some sheep moved between the red button and the food shoot, alternately using the button and a food ball. Other sheep would use the button two or three times then use (eat) two or three food balls at once. Yet another sheep wedged itself on the shoot so that it could reach and use the food machine button while allowing the food to effectively roll into its mouth. It could use the food machine and food without needing to move at all. These different behavioural responses are illustrated in Figure 8(b), (c) and (d) respectively.

A number of other tasks designed for the sheep are shown in Figure 9. These range from static, unscripted objects such as a pyramid, to scripted objects such as a colour changing screen and height changing wall. Behaviours learned include changing the colour of the screen from red to green and increasing and decreasing the height of the wall. The ability of the sheep to learn behaviours associated with each of these different artefacts, built at different times and potentially by different players, illustrates the flexibility of the CFG representation to facilitate reasoning about objects and avatars without prior knowledge of what these may be.

6. DISCUSSION

The ability of NPCs using MRL to learn behaviours which adapt to new objects and avatars in their environment makes possible a new generation of games in which players can make open-ended modifications to the game environment while the game is in progress. In our sheep herding game, sheep using MRL with CFGs are able to respond to modifications to their environment even though the exact changes made by players or the times and places they will be made are not known in advance. The behavioural responses of each sheep are based on their experiences in their environment. Because the experiences of each sheep are different, their responses to similar situations can vary, creating a cast of different characters.

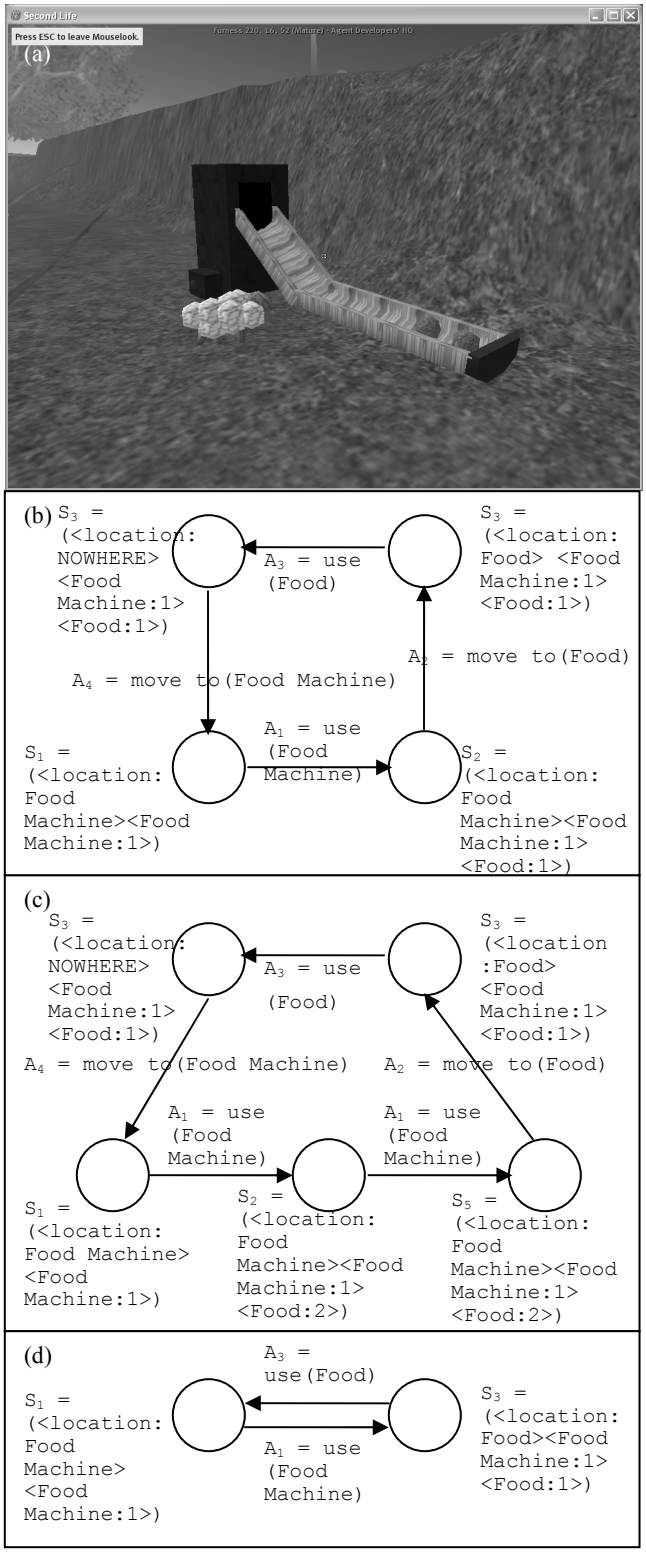

Figure 8. (a) A sheep explores the food machine. (b) A learned behavioural cycle for eating one food ball at a time. (c) A learned behavioural cycle for eating two food balls at a time. (d) A learned behavioural cycle for obtaining and eating food without moving.

Figure 9. Other tasks learned by sheep: changing the colour of a screen, changing the height of a wall, jumping through a hoop to a target.

While the demonstration in the previous section gives an indication of the adaptability of MRL agents controlling NPCs, a number of constraints imposed by the Second Life virtual environment do limit this demonstration. The speed of the simulation is an issue, with a three second server side delay on XML-RPC calls slowing the simulation considerably. A limit of 256 characters per message sent between the client application and the Second Life server also limits the number of properties of each object that can be sensed. The CFG representation is flexible enough to allow different objects to be defined in terms of different properties, however in this demonstration we were limited to sending only the name of sensed objects between the client and server.

Despite these limitations, which we envisage will disappear with the improvement in virtual world technologies, the sheep herding game demonstrates the key concepts of a new kind of open-ended computer game. Players have a set of open-ended modelling tools with which they can modify the game environment and characters are able to reason about and respond to these changes.

The nature of the game presented in this paper is such that each game play sequence is different. This makes a statistical analysis of the performance of the MRL agents difficult. Instead, this paper used finite state automata diagrams to illustrate the adaptable behavioural cycles learned by MRL agents. Other work [8] has considered MRL in controlled, simulated game scenarios and empirically evaluated the variety and complexity of learned behaviours to provide an insight into the adaptive, multi-task learning ability of MRL. This work also illustrates the applicability of MRL to game scenarios beyond this paper.

7. CONCLUSION

This paper presents a model for adaptive non-player characters in open-ended virtual worlds. These worlds provide a platform for a new generation of computer games in which players can modify the game environment using open-ended modelling tools. This paper identifies a need to develop characters that can respond autonomously to the unpredictable, open-ended changes to their environment as a key challenge in the development of such games. We present MRL using CFGs as a means of representing unpredictable, evolving worlds for character reasoning and demonstrate the capabilities of these characters in an open-ended sheep herding simulation game implemented in Second Life.

This game extends existing simulation games with the ability for players to creatively modify the game world using open-ended modelling tools while the game is in progress. The ability of the sheep to learn behaviours associated with each of these different artefacts, built at different times and potentially by different players, illustrates the flexibility of the MRL algorithm and CFG representation to facilitate reasoning and learning about objects and avatars without prior knowledge of what these may be.

8. ACKNOWLEDGMENTS

This research was supported by a National ICT Australia PhD scholarship. National ICT Australia is funded by the Australian Government's Backing Australia's Ability initiative, in part through the Australian Research Council.

This paper was written while Mary Lou Maher was working at the National Science Foundation in the United States. Any opinion, findings and conclusions or recommendations expressed in this chapter are those of the authors and do not necessarily reflect the views of the National Science Foundation

9. REFERENCES

[1] Activeworlds. "Activeworlds". www.activeworlds.com (Accessed January 2007).

[2] Johnson, D. and Wiles, J. "Computer games with intelligence", presented at *The 10th IEEE International Conference on Fuzzy Systems*, pp1355-1358, 2001.

[3] Laird, J. and van Lent, M. "Interactive computer games: human-level AI's killer application", presented at *National Conference on Artificial Intelligence*, pp1171-1178, 2000.

[4] Linden. "Second Life", www.secondlife.com (accessed January, 2007).

[5] Maher, M.-L. and Gero, J.S. "Agent models of 3D virtual worlds", ACADIA 2002: Thresholds. California State Polytechnic University, Pamona, pp. 127-138, 2002.

[6] Merceron, A. *Languages and Logic*: Pearson Education Australia, 2001.

[7] Merrick, K. and Maher, M. L. "Motivated reinforcement learning for non-player characters in persistent computer game worlds" presented at *ACM SIGCHI International Conference on Advances in Computer Entertainment Technology*, 2006, CA, USA, (CD, no page numbers), 2006.

[8] Merrick, K. 2007, "Modelling Motivation for Experience-Based Attention Focus in Reinforcement Learning", PhD Thesis, University of Sydney (manuscript).

[9] Nilsson, N. J. *Introduction to machine learning*: http://ai.stanford.edu/people/nilsson/mlbook.html (accessed January, 2006), 1996.

[10] Rex, F. "LambdaMOO: An introduction", http://www.lambdamoo.info, (accessed December, 2006)

[11] Sutton, R. S. and Barto, A. G. *Reinforcement learning: an introduction*: The MIT Press, 2000.

[12] Woodcock, S. "Games making interesting use of artificial intelligence techniques" http://www.gameai.com/games.html (accessed October 2005)